Uncovering the Potential of Contrastive Predictive Coding: A Comparative Study of Transformers and ResNets

Nick Hamann, Jared Gage, Anjali Singh, Ayush Kumar
Georgia Tech
nhamann3@gatech.edu, jgage6@gatech.edu, asingh972@gatech.edu, akumar886@gatech.edu

Abstract

Data-efficient encoding of visual data is important when labeled data can be computationally expensive and time-consuming to obtain. While there are several methods for efficient encoding, none use the Transformer architecture for image recognition. The current study proposes an efficient ViT paradigm that outperformed most others in accuracy and amount of labeled data required. Modern ResNet architectures also improved on previous results. Various data augmentation techniques are discussed.

1. Introduction

One of the biggest challenges in many deep learning endeavors is a shortage of labeled data. Acquiring enough data for larger models to be properly trained can be financially and temporally expensive. Consequently, methods to reduce the required amount of training data are increasingly valuable.

One method for reducing the amount of necessary training data in deep learning applications is called Contrastive Predictive Coding (CPC). CPC is a self-supervised learning method of using past samples to predict future values. In the field of speech recognition, this means using previous clips to predict what will be said next and then comparing to what is actually said. One of the main works in this space was produced by van den Oord et al. in 2018. [18] This paper uses CPC for audio prediction and proposes a loss function called InfoNCE which increases training efficiency. Noise contrastive estimation (NCE) finds positive and negative samples in regard to the model's prediction and uses these to filter out noise.

Henaff et al. present a similar study in the field of computer vision. [5] The researchers use random crops of an image to create an encoded representation of classes. The representations are created by training a model to distinguish between crops of the same image and crops of a random different image, which can be referred to as positive and negative crops. After these representations have been created, a ResNet is used to classify these encodings, and at the time was able to achieve state-of-the-art performance on image classification tasks given a small amount of training data.

"Attention is All You Need" is a landmark study in neural network architecture that was produced in 2017. [4] The work proposed the modern transformer model, which has become ubiquitous in high-performing neural network architectures. Transformers and CPC were combined into a concept called CPCTR (CPC Transformer) in a recent paper using this new framework for video classification tasks. [8] Specifically, the encoder is modified to be a vision transformer model (ViT), which improves the ability to create abstract representations. Though the authors don't focus on improving the efficiency of training, the results were state-of-the-art in video classification.

The purpose of the current study is to reproduce the results of some previous studies and attempt to improve upon their results. Piecing together some of the previous studies, we wish to use a transformer model to improve on the results of Henaff et al in image classification. [5] The aim is to continue the positive results with low amounts of training data while increasing the accuracy. As this study was produced several years ago, the state of the art has improved, and results are expected to improve. Even small improvements are important in this space and will reduce expenses incurred by training models, allowing greater accuracy with fewer examples.

The impacts are potentially far-reaching. Smaller, faster training pipelines that use a fraction of the data other models require could run better on mobile devices, allowing models to be useful in settings that were previously impractical. Further, research is being done into the environmental impacts of some of the large models that are currently being commercialized and consumerized. [6] As models continue to follow the trend of increasing in size, more efficient training will be an effective way to mitigate some of this potential harm.

Rather than solving ImageNet as in the CPC papers,

which has 1.2 million training images, we used the CIFAR-10 dataset. This is a collection of 60,000 labeled 32x32 images belonging to 10 classes, with 600 examples of each class. This was chosen for its complexity, and also because its small size allowed for faster training. Furthermore, while some datasets may have tens of thousands of examples of each class, this only has 6000, making it a good potential candidate for a model that needs to be trained to efficiently classify.

2. Approach

We used PyTorch to implement Contrastive Predictive Coding (CPC) in the style of [5] and [18] for multiple different encoder and classifier networks, and evaluated these different encoders in both the standard linear evaluation protocol and the efficient classification experiments from [5].

In CPC, (using the notation from [5]), an input image $\mathbf{x}$ is preprocessed into a grid of overlapping sub-images $\mathbf{x}_{i,j}$, and mapped by the encoder network f_θ to produce a grid of vectors $\mathbf{z}_{i,j} = f_\theta(\mathbf{x}_{i,j})$. During self-supervised learning, these latent representations are mapped by the context network g_ϕ to produce a grid of context vectors $\mathbf{c}_{i,j}$.

Specifically, we implemented encoder networks based on ResNet-v1 [10] and ResNet-v2 [11], as well as ViT [8]. We also implemented the image preprocessing and training code for all 3 learning setups.

One barrier to reproducing results was the lack of source code provided with either of [5] and [18], though two snippets of code are provided in the appendix of [5]. One open source implementation for CPC which was utilized and modified for the ResNet-v1 encoder was [15], though it is a partial implementation as it implements linear classification only and not the efficient classification experiments.

Oord et al. [18] introduce the InfoNCE loss as a way to learn from unlabeled data. Our implementation was loosely based on the implementation in [15]. This loss function takes a patch of an image and compares it with surrounding patches, and also unrelated patches from other images. In practice, this loss should be smallest when patches are from the same image and largest when they are from different images.

For all experiments, the context network used is the implementation from [9], which is adapted from a code snippet provided in the appendix of [5]. We did attempt to implement PixelCNN ourselves, following [17]. We obtained good results on grayscale datasets like MNIST and Fashion-MNIST, as tested by using the PixelCNN as a generative model and sampling images from it, but we were unable to get it to produce realistic images for RGB datasets like CIFAR-10, so we opted to use the implementation mentioned above.

In addition to the ResNet encoder implementations, we also implemented a version of CPC using a Vision Transformer for the encoder net, seeking to understand the effects of self-attention on self-supervised learning using CPC. We anticipated that ViTs might need a vast amount of data and might not perform well on CIFAR-10. However, because the problem was framed in a self-supervised manner, the amount of positive and negative samples gave us on-par results from the baseline. We were also uncertain about generating patched embeddings from pre-sampled patches of the images that were already small having a size of 32x32 (CIFAR-10). Tuning of patched embedding size was a challenge as giving a single patch as a whole would not help the self-attention mechanism and dividing the image patch per pixel also wouldn't help the training. In our case, a ViT pixel size of 4 worked the best.

The source code is located here.

3. Experiments and Results

3.1. Data Efficient Encoding with ResNet-V2 Encoder

3.1.1 Self-supervised learning

Since [18] uses a ResNet-101 with ResNet-v2 blocks (a.k.a. PreAct residual blocks) as in [11] for its encoder network, we adopt a similar architecture, using blocks defined by

$$\text{Sublayer}(x) = \text{Conv}(\text{ReLU}(\text{Conv}(\text{ReLU}(x)))) \quad (1)$$

$$\text{PreActBlock}(x) = \text{Skip}(x) + \text{Sublayer}(x) \quad (2)$$

where Conv is 3x3 with a padding of 1, and Skip is either the identity or a 1x1 convolution (for when there are different numbers of input and output feature maps).

For this residual block, we do not use normalization layers. In tuning experiments, we found that using batch normalization layers in the encoder ResNet harms training, matching results obtained in [5]. Henaff et al. hypothesize that using batch normalization harms the self-supervised CPC training because it "introduces a dependency between patches (through the batch statistics) that can be exploited", however they report a performance improvement by using layer normalization [1] instead. In our experiments, we also tried various arrangements of Layer Normalization with both the PreAct ResNet block and also a ResNet-v1 version, but the models always converged to a fixed InfoNCE loss of around 4.5 and did not change with further training. Therefore, we opted to use no normalization in the encoder.

We use a ResNet-18 [10] encoder for CIFAR-10: 4 layers, 2 blocks each, where the respective layers have (64, 128, 256, 512) feature maps.

We study 3 different augmentation approaches:

- A_1 = (random horizontal flip w/ $\mathbb{P} = 0.5$)

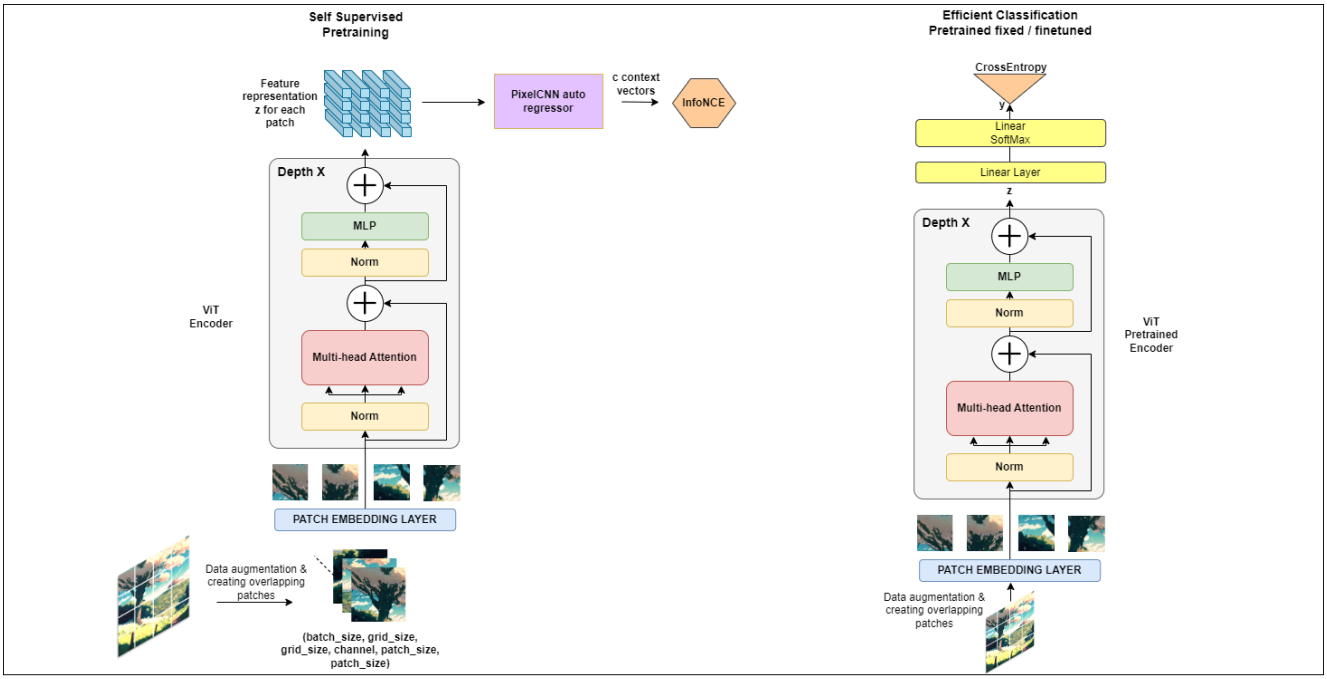

Figure 1. Architecture of semi-supervised learning with Contrastive Predictive Coding using Vision Transformers. Left: Self-supervised pre-training with spatial prediction task using InfoNCE loss. For each image, augmentation transforms are applied, and it is divided up into a grid of overlapping patches. Now to create a global representation of these patches, the patches are again subdivided into smaller non-overlapping patches by introducing a patched embedding layer of the Vision transformer. Thus, for each patch, we create a feature vector z (blue). These feature vectors above a certain place in an image are aggregated as context vector c which is used to predict the next feature vectors. The InfoNCE loss is used to train this self-supervised setup. Right: The pre-trained encoder from the self-supervised setting is used for the classification task by replacing the context network with two linear layers and cross-entropy loss. The network is trained with two methods: (A) freezing the encoder weights and just training the linear layers with the given subset of labeled data, and (B) fine-tuning the full network with the given subset of labeled data.

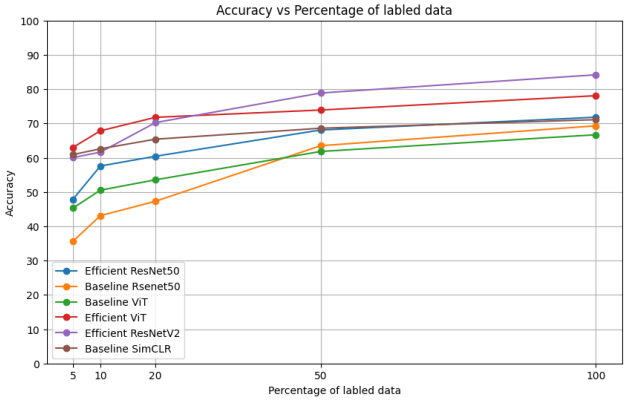

Figure 2. Accuracy vs Percentage of labeled data

- $A_2 = A_1$ + (random conv. to grayscale w/ $\mathbb{P} = 0.25$)
- $A_3 = A_2$ + (random color jitter)

For the InfoNCE loss, we used a larger value of $N = 100$ for the hyperparameter N = the number of negative samples, Section 2.3 and Appendix A of [18] show that minimizing the InfoNCE loss maximizes a lower bound on the mutual information between the context vector $\mathbf{c}_t$ and the latent vector $\mathbf{z}_{t+k}$, a lower bound that gets tighter as N, the number of negative samples, gets larger.

We follow [5] in training the self-supervised phase using Adam as the optimizer. Our learning rate, 5e-4, which seemed to work better during tuning experiments, is close to the 4e-4 used by Henaff et al.

3.1.2 Linear classification

We use the same linear evaluation protocol described in [5]: take a ResNet-18 encoder network trained through self-supervision (via the InfoNCE loss), freeze its weights, add a single linear classifier layer at the end, and train on 100% of the training set.

We train with as large a batch size as fits in memory, and on our RTX 3060, this is 128. We train for 15 epochs, using the Adam optimizer with the default learning rate of 1e-3.

Augment	Accuracy
A_1	63.4
A_2	65.6
A_3	62.5

Table 1. Linear evaluation protocol test accuracies on CIFAR-10. The encoder network weights are frozen. There are different augmentation transforms A_1, A_2, and A_3. For all experiments, we use a PreReact (ResNet-v2) ResNet-18 encoder and a single linear layer + softmax.

	2%	5%	10%	20%	50%	100%
A_1	49.5	60.0	61.6	70.2	78.9	84.2
A_2	**53.4**	**63.1**	**66.4**	71.3	**80.6**	**85.1**
A_3	49.2	58.8	64.5	**72.8**	78.8	84.8

Table 2. Test accuracies on CIFAR-10 for the efficient classification problem with different augmentation transforms A_1, A_2, and A_3. For all experiments, we use a PreReact (ResNet-v2) ResNet-18 encoder and a PreReact ResNet-14 classifier.

3.1.3 Efficient classification

In this experimental setup, we study the effect of self-supervised learning has for improving learning from small amounts of labeled data. We take a learned ResNet-18 encoder network, append a ResNet-14 classifier, and then train on fractions of CIFAR-10: (2%, 5%, 10%, 20%, 50%, 100%).

For this classification setup, there are two main differences from the linear setup: first, as in [5], we don't freeze the encoder weights, and second, instead of a linear classifier, we use a (PreAct) ResNet-14 as the classifier, consisting of 3 layers of 2 blocks each, with all layers having 512 feature maps. We do use batch normalization in the classifier ResNet, finding in tuning experiments conducted on both CIFAR-10 and CIFAR-100 that it adds +1-2% validation accuracy.

We again train with as large a batch size as fits in memory, and on the same device as before the batch size is now 64. We train for 15 epochs using the Adam optimizer. This differs from [5], which uses SGD and dynamically varies the learning rate in conjunction with other parameters, such as the weight decay, the drop out, and the per-worker batch size. For simplicity, we just use Adam with an initial learning rate of 5e-5, the same that [5] uses for their linear evaluation case.

3.1.4 Results and Discussion

The results for CPC with the ResNet-v2 encoder on the linear classification and the efficient classification problems are shown in the tables 1 and 2.

The A_2 augmentation, which differs from A_1 by randomly converting 25% of the dataset to grayscale, is uniformly better than the baseline A_1 in all experiments by about +1% test accuracy. In A_3, we add color jitter on top of the grayscale conversion added in A_2, which seems to slightly decrease the performance. This is, on the surface, unexpected, since color jitter is one of the transformations applied in SimCLR [7]. However, this may be a result of our having applied color jitter to all images in the training set, as opposed to what they do in [5], which applies extra transformations to only 80% of the data.

The resulting accuracies are in the 62-66% range, which is poor, most likely a result of some combination of not using enough data augmentation and not training in the self-supervised phase. In [16], they report a network trained for 100 epochs with a ResNet-18 backbone that achieves 84% accuracy (87% after 300 epochs) on the linear evaluation protocol, significantly higher than the accuracies achieved here.

3.2. Data Efficient Encoding with ResNet Encoder

3.2.1 ResNet-v1 Vs ResNet-v2

As an experiment, we also tried to use ResNet-v1 as the encoder to compare with the ResNet-v2 in a self-supervised setting, which was originally used in the paper Henaff et al [5]. We found that ResNet-v2 outperformed ResNet-v1 even with a smaller architecture. By just using 18 convolution blocks in ResNet-v2, it outperformed ResNet-v1 with 50 convolution blocks on the Linear and Efficient Classification task. The accuracy comparison results for Efficient classification can be seen in the figure 2 or in the table 3 given below.

The ResNet-50 consists of 4 layers having [3,4,6,3] blocks respectively, where each of the blocks consists of 3 convolution layers with (64,128, 256, 512) feature maps. There are also 2 initial convolution layers which make a total of 50 convolution layers. The implementation of ResNet-v1 is based on [13].

The primary difference between ResNet-v1 and ResNet-v2 is how they handle skip connections. In ResNet-v1, these skip connections are simply added to the output of the corresponding convolutional layer. In ResNet-v2, the skip connections include additional ReLU activation functions, before they are added to the output of the convolutional layer. The ResNet-v2 can help to reduce the impact of "internal covariate shift", which is the phenomenon where the distribution of the activations in a layer changes as the input to the layer changes during training.

3.2.2 Training Process and Results

In the self-supervised learning setting the model is trained in a similar fashion as described in Henaff et al [5]. The encoding representation was taken from the last layer of the

	5%	10%	20%	50%	100%
ResNet-v1	47.87	57.58	60.43	68.08	71.81
ResNet-v2	60.03	61.62	70.24	78.87	84.18

Table 3. Test accuracies on CIFAR-10 for the efficient classification ResNet-v1 compared to the performance of ResNet-v2.

ResNet after removing the fully-connected layer. The objective was to minimize the InfoNCE loss by taking positive samples from current images and negative samples from the rest of the batch. It was trained on Google Colab which provides TeslaK80 GPU. The model was trained with a batch size of 64 and with a learning rate of 1e-3 for 19 epochs. The number of negative samples was 100.

Once we have the self-supervised model trained we used that for linear classification similar to the ResNet-v2 by adding a linear layer to the encoder output.

For the "Efficient classification", the model is trained in a similar way as ResNet-v2.In this setting, the model was trained with a batch size of 128, a learning rate of 1e-3 for 10 epochs on CIFAR-10 with(5%,10%,20%,50%,100%) of the labeled data. The results are shown 2 with comparisons with other models and in table 3.

We also tried to see the effect of increasing the number of negative samples size on efficient classification. Theoretically, increasing the number of negative samples should increase the accuracy. As the model gets more negative examples, it tries to separate the positive examples from more newly added negative samples which increases the overall performance of the model. But in our case, when we doubled the negative sample size the accuracy decreased from 71.81% to 70.46% on 100% CIFAR on efficient classification. A larger number of samples may be able to achieve lower losses than fewer samples if the models are trained for more epochs.

3.3. Data Efficient Encoding with Vision Transformer

3.3.1 Training setup

The setup for self-supervised training with a ViT encoder is shown in Figure1. Here, the CNN-based encoder is replaced with ViT encoder. The representation vector of each patch is generated by first creating non-overlapping sub-patches using the patch embedding layer. The patched embedding are then passed through the transformer blocks which contains multi-headed attention layer, norm, and feed-forward layers. Our implementation of ViT is based on [12] Thus, for each patch we create a feature vector z. These feature vectors above a certain place in an image are aggregated as context vector c which is used to predict the next feature vectors using a PixelCNN network as the context network. The InfoNCE loss is used to train this self-supervised setup. The pretrained encoder from the self-supervised setting is then used for the classification task by discarding & replacing the context network with two linear layers and cross-entropy loss. The network is trained using two methods: (A) Freezing the encoder weights and only training the linear layers with a given subset of labeled data. (B) Fine-tuning the full network with a given subset of labeled data. We experimented with different variations of linear layers while fine-tuning the ViT encoder where we got best results by adding two linear layers. We performed hyperparameter tuning with our ViT encoder to improve performance on the CIFAR-10 where we got the best results at a learning rate of 1e-4 as increasing learning rate to 1e-3 was overshooting the loss., batch size of 64. The best complexity of the model had ViT patch size of 4, number of attention heads 8, embedding dimension 512. Reducing the depth of transformer network from 12 to 6 gave the best performance on test set without overfitting the data. The training was done using Google Colab Tesla K80 GPUs.

3.3.2 Results

In our comparative study, we found that in low labeled settings the ViTs did outperform CNN based models with Top 1% accuracy of 62.99%, 67.85% and 71.78% with 5%, 10% and 20% labeled data respectively (see Figure 2). This data efficiency of ViTs was counter intuitive because the transformers models are known to require a large amount of training data due to their weaker inductive bias. As we keep increasing the labeled data size the performance gap between ViT and ResNetV2 encoder decreased and at 100% data the ResNetV2 performed better. This can be due to the fact that the ViT encoder was not trained long enough (only up to 30 epochs) due to resource constraints. The self-supervised ViT outperformed baseline supervised learning ViT with accuracy of 76.16% against 66.7% on 100% labeled data which proves the effectiveness of CPC-based training (see Table 4).

3.4. Comparison with other state-of-the-art frameworks: SimCLR

In order to better validate the results of the transformer encoder, other recent methodologies were used as a benchmark. Specifically, "a simple framework for contrastive learning of visual representations" (SimCLR) was chosen for its high performance in the space of efficient classification. Similar to the CPC models, SimCLR relies on comparing two images, though it performs different data augmentations on the same image and then tries to maximize a similarity score, while CPC uses positive and negative patches of different images.

A full pipeline was created to train a SimCLR model, and some hyperparameter tuning took place. Using a ResNet-

	5%	10%	20%	50%	100%
Fixed ViT					68.1
Baseline ViT	45.3	50.5	53.6	61.8	66.7
Efficient ViT	**62.9**	**67.9**	**71.8**	**73.9**	**78.1**
SimCLR	60.9	62.6	65.4	68.6	71.1

Table 4. Test accuracies on CIFAR-10 for the efficient classification vision transformer compared to the performance of a baseline SimCLR encoder.

18 trained for 20 epochs, the preliminary results were poor. Ultimately, it was decided to use a pre-trained ResNet-50 model that got results better aligned with the SimCLR paper [7]. As this was meant to be a benchmark for the results of the efficient vision transformer, it did not make sense to compare with a model that was far inferior to its potential. The pretrained SimCLR implementation was used to produce labels for the augmented dataset, and a logistic regression model was then trained on the data to determine the performance of the SimCLR encoder's representations. There was mild overfitting in the regression model, with training and validation accuracies differing by as much as 20%, so the learning rate was decreased to slow this overfitting. This was trained using Adam optimizer with a learning rate of 1e-4.

One other point to be made was that SimCLR was found to benefit from larger batch sizes. [7] However, since the current approach is meant to efficiently classify images, it was decided to use a smaller batch size of 64, aligned with the batch sizes that the other ResNet encoders were using. In very small dataset sizes, it may be impractical to have large batch sizes, so for efficient classification purposes a smaller batch size seemed appropriate.

The results of the comparison can be seen in Table 4. While the SimCLR encoder produced higher accuracy scores than the ResNet-50 and base ViT, the efficient ViT outperformed the SimCLR results in each efficient trial.

4. Discussion and Future Direction

Data-efficient vision transformers were successful and outperformed other near-state-of-the-art methods for efficient image classification. The self-supervised pretrained transformers can be tuned on very less data for downstream tasks decreasing the time for training. However, transformers are still computationally expensive models, which have other drawbacks. As our future direction of this research, we can look at computationally efficient transformers like MobileViT and DeiT etc. We can try distillation techniques to reduce the parameter size of our self-supervised ViT with a teacher-student strategy or pruning to reduce the parameters for downstream tasks.

One limitation of our study is reporting test accuracies for single runs, rather than averaging runs over multiple random seeds, something we'd ideally like to do provided we had a larger compute budget.

For each full run of ResNet-v2 (self-supervised + linear + efficient), it took more than 17 hours on the hardware used (a single RTX 3060). If we had more time, it would be interesting to also train the CPC w/ ResNet-v2 encoder to 100 or even 300 epochs and compare the change in accuracies, however this would take up to 90 hours of training in total (an additional 81 hours) on the same device.

Large amounts of compute seems to be a common theme with contrastive learning, including in CPCv1 [18], where they train their encoder on ImageNet using 32 GPUs, and in [5], where they train for 200 epochs. In [14], a tutorial on implementing and training SimCLR, they remark that:

> [C]ontrastive learning benefits a lot from long training. The shown plot above is from a training that took approx. 1 day on a NVIDIA Titan-RTX. ... In general, contrastive learning can also benefit from using larger models, if sufficient unlabeled data is available.

We trained on the self-supervised phase for only 9 hours (30 epochs), rather than a full day, in order to facilitate more experiments. Therefore, higher accuracies should be possible with longer training.

Long training times are one limitation of the method studied here: there are scenarios where computational resources are readily available but necessary amounts of labeled data are impossible or impractical to obtain, but ultimately it would be desirable to have methods that are both data- and compute-efficient.

Future work would probably want to also focus on fully implementing the CPCv2 augmentation steps is especially the elastic deformation and shearing transformations, as well as the full suite of color augmentations, including fixing color jitter. The CPC implementation [15] includes a set of 14 different transformations, and two random transformations are applied to each patch during augmentation. Fully implementing this suite of transformations would be an obvious next choice for improving model performance.

Another future direction could be exploring alternative datasets to train on during the self-supervised phase, for example pre-training on STL-10 [3], and then performing the same linear and efficient classification experiments on CIFAR-10. STL-10 images are larger (96x96 rather than 32x32), and there is more training data available for self-supervised learning (100,000 versus 50,000). Larger datasets have been observed to be beneficial for contrastive learning approaches like CPC [2].

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[2] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023. 6

[3] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011. 6

[4] Ashish Vaswani et al. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[5] Oliver Henaff et al. Data-efficient image recognition with contrastive predictive coding, 2020. PMLR. 1, 2, 3, 4, 6

[6] Pengfei Li et al. Making ai less 'thirsty': Uncovering and addressing the secret water footprint of ai models. *arXiv preprint arXiv:2304.03271*, 2023. 1

[7] Ting Chen et al. A simple framework for contrastive learning of visual representations. *International conference on machine learning*, 2020. 4, 6

[8] Yue Liu et al. Contrastive predictive coding with transformer for video representation learning. *Neurocomputing*, 482:154–162, 2022. 1, 2

[9] William Falcon. Lightning bolts - pixelcnn implementation. https://github.com/Lightning-Universe/lightning-bolts/blob/master/pl_bolts/models/vision/pixel_cnn.py. 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. pages 630–645, 2016. 2

[12] JackKoLing. Vit-pytorch. https://github.com/JackKoLing/VTCC. 5

[13] JayPatwardhan. Resnet-pytorch. https://github.com/JayPatwardhan/ResNet-PyTorch. 4

[14] Phillip Lippe. Tutorial 17: Self-supervised contrastive learning with simclr. https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial17/SimCLR.html. 6

[15] Ronen Schwarz. Cpcv2-pytorch. https://github.com/rschwarz15/CPCV2-PyTorch. 2, 6

[16] Yuandong Tian. Understanding deep contrastive learning via coordinate-wise optimization, 2022. 4

[17] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks, 2016. 2

[18] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint*, 2018. arXiv:1807.03748arXiv:1807.03748. 1, 2, 3, 6